CERTIFICATE OF MAILING/TRANSMISSION
I hereby certify that this correspondence is being submitted electronically via EFS Web on the date shown below.

/Rob Brownstein/ October 26, 2009

Rob Brownstein *Date*

Attorney Docket No.: 42P16521 *Patent*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	)	
	)	
Gates, et al.	)	Examiner: Satish Rampuria
	)	
Application No.: 10/676,311	)	Art Unit: 2191
	)	
Filed: September 30, 2003	)	Confirmation No.: 8182
	)	
For: GENERATING EXECUTABLE CODE BASED ON CODE PERFORMANCE DATA	)	
	)	

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF – 37 C.F.R. § 41.37 (CORRECTED)

Sir/Madam:

This Corrected Appeal Brief is submitted pursuant to the Notification of Non-Compliant Appeal Brief issued on Sept. 30, 2009 for the above-captioned patent application. The Notice of Appeal and this Appeal Brief are filed in response to the Final Office Action mailed June 8, 2009. Appellants respectfully request consideration of this appeal and allowance of the application by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is Intel Corporation ("Intel"), a Delaware corporation having a principal place of business at 2200 Mission College Blvd., Santa Clara, California, 95052. Intel is the assignee of the entire right, title and interest in the above-captioned application by virtue of an assignment recorded at the U.S. Patent Office at Reel 014959, Frame 0172.

II. RELATED APPEALS AND INTERFERENCES

Appellants and Appellants' legal representative know of no interferences, appeals, or other proceedings that will directly affect, be directly affected by, or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1, 3-10, 16-20, 41, 43-45, 50, and 53 are **pending** in the application and are the **claims on appeal**. Claims 2, 11-15, 21-40, 42, 46-49, and 51-52 are **cancelled**. All claims in the application currently stand rejected based on U.S. Patent No. 6,289,505 B1 to *Goebel* (hereinafter "*Goebel*"), U.S. Patent Publication No. 2003/0051234 A1 to *Schmidt* (hereinafter "Schmidt"), U.S. Patent No. 6,874,140 B1 to *Shupak* (hereinafter "Shupak"), U.S. Patent No. 7,000,227 B1 to *Henry* (hereinafter "Henry"), and allegedly Admitted Prior Art. The basis of rejection of the claims is as follows:

(i) Independent claims 1, 18, 41, and 50 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Goebel in view of Schmidt.

(ii) Dependent claims 3-10, 16, 17, 19, 20, 43-45, and 53 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Goebel in view of Schmidt.

IV. STATUS OF AMENDMENTS

As of the Final Office Action mailed June 8, 2009, claims 1, 3-10, 16-26, 29-31,

41, 43-47, 49, 50, and 53 were pending in the application. Appellants filed an Amendment After Final on August 10, 2009 cancelling claims 21-26, 29-31, 46, 47, and 49 to place the application in better form for appeal. On August 11, 2009 Appellants filed a Notice of Appeal in response to the Final Office Action. Therefore, the claims on appeal are claims 1, 3-10, 16-20, 41, 43-45, 50, and 53. None of these claims have been amended subsequent to final rejection.

A copy of all claims on appeal, as finally rejected on June 8, 2009, is attached hereto in Appendix A.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Embodiments of the present invention, as recited in independent claim 1, relate generally to a method for improving compiler-generated executable code. (e.g., see para. [0009], line 1) To improve the executable code, source code (e.g., source code 310, para. [0031], line 2; element 402 of FIG. 4) is received and transformed into intermediate code. (e.g., intermediate code 322, para. [0031], line 3; element 404 of FIG. 4) The intermediate code is executed based on external execution input. (e.g., external execution input 332, para. [0019], lines 5-6; FIG. 3a) Data indicating performance of the intermediate code is generated (e.g., element 410 of FIG. 4; para. [0033], lines 1-3) when the intermediate code is executed with the external execution input. (e.g., FIG. 3a; para. [0019], lines 5-6) Machine code is produced based on the data and the intermediate code. (e.g., FIG. 3a; para. [0019], lines 7-8)

Embodiments of the present invention, as recited in independent claim 18, relate generally to a method for improving compiler-generated executable code. (e.g., see para. [0009], line 1) To improve the executable code, source code (e.g., source code 310, para. [0031], line 2; element 402 of FIG. 4) is transformed into intermediate code. (e.g., intermediate code 322, para. [0031], line 3; element 404 of FIG. 4) The intermediate code is provided to a profiler (e.g. profiler 330, para. [0019]; FIG. 3a) that executes the intermediate code based on external execution input (e.g., external execution input 332, para. [0019], lines 5-6; FIG. 3a) and generates annotated intermediate code (e.g. annotated intermediate code 336, para. [0019], line 7) based on the performance of the

executed intermediate code when the intermediate code is executed with the external execution input. Annotated intermediate code is received from the profiler. (See para. [0025], lines 4-5) The annotated intermediate code is transformed into machine code. (See para. [0025], lines 5-6)

Embodiments of the present invention, as recited in independent claim 41, relate generally to an article of manufacture for improving compiler-generated executable code. (e.g., see para. [0009], line 1) To improve the executable code, instructions on a computer readable storage medium are executed and cause an electronic system to receive source code, (e.g., source code 310, para. [0031], line 2; element 402 of FIG. 4) produce intermediate code based on the source code, (e.g., intermediate code 322, para. [0031], line 3; element 404 of FIG. 4) and execute the intermediate code based on external execution input. (e.g., external execution input 332, para. [0019], lines 5-6; FIG. 3a) The instructions on the computer readable storage medium are also executed to cause the electronic system to generate performance data indicating performance of the intermediate code (e.g., element 410 of FIG. 4; para. [0033], lines 1-3) when the intermediate code is executed with the external execution input (e.g., FIG. 3a; para. [0019], lines 5-6) and produce machine code based on the intermediate code and the performance data. (e.g., FIG. 3a; para. [0019], lines 7-8)

Embodiments of the present invention, as recited in independent claim 50, relate generally to a system for improving compiler-generated executable code. (e.g., see para. [0009], line 1) The system includes a processor (e.g. processor 120, para. [0012], line 3) coupled to a dynamic random access memory, (e.g. memory 130, para. [0012], line 3; para. [0014], line 7) and a computer readable storage medium with sequence instructions for execution. (para. [0014], line 6) To improve the executable code, the system executes the sequence of instructions that cause an electronic system to receive source code. (e.g., source code 310, para. [0031], line 2; element 402 of FIG. 4) Source code is received and transformed into intermediate code. (e.g., intermediate code 322, para. [0031], line 3; element 404 of FIG. 4) The intermediate code is executed based on external execution input. (e.g., external execution input 332, para. [0019], lines 5-6; FIG. 3a) Data indicating performance of the intermediate code is generated (e.g., element 410 of FIG. 4; para. [0033], lines 1-3) when the intermediate code is executed with the external

execution input. (e.g., FIG. 3a; para. [0019], lines 5-6) Machine code is produced based on the data and the intermediate code. (e.g., FIG. 3a; para. [0019], lines 7-8)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The issues presented in this appeal are:

(i) Whether independent claims 1, 41, and 50 and dependent claims 3-10, 16, 17, 43-45, and 53 are obvious in view of Goebel and Schmidt under 35 USC § 103(a).

(ii) Whether independent claim 18 and dependent claims 19 and 20 are obvious in view of Goebel and Schmidt under 35 USC § 103(a).

VII. ARGUMENTS

This section sets forth Appellants' arguments against the rejections and in favor of the patentability of the claims on appeal. Part VII.A provides a brief overview of the Goebel and Schmidt references that the Final Office Action relied upon to reject the claims on appeal. Part VII.B discusses why the combination of Goebel and Schmidt cannot render independent claims 1, 41, and 50 and dependent claims 3-10, 16, 17, 43-45, and 53 obvious. Part VII.C discusses why the combination of Goebel and Schmidt cannot render independent claim 18 and dependent claims 19 and 20 obvious.

A. Brief Overview of Goebel and Schmidt.

1. Goebel

Goebel discloses methods for re-optimizing a binary executable. *Goebel,* col. 3, lines 39-41. Part of optimizing the binary executable includes an intermediate representation optimizer segment 305 that processes profile information 317 to determine which portions of the binary executable most need to be optimized. *Goebel,* col. 6, lines 61-66; FIG. 3. The profile information 317 or "execution profile" is "a collection of data, gathered while the program executes ... that can be used to analyze the program's

performance." *Goebel,* col. 4, line 66 to col. 5, line 4. The profile information is generated during execution of the binary executable. *Goebel,* col. 6, lines 63–66; col. 7, line 66 to col. 8, line 2.

2. Schmidt

Schmidt discloses a front end complier generating unique instructions for virtual method calls in intermediate representation code that may be passed to a back-end compiler. *Schmidt,* Abstract. FIG. 6 illustrates a front-end complier 610 processing source code 105, and generating an intermediate representation 615 that is fed to the back-end compiler 620. *Schmidt,* para. [0040], lines 1-5. The back-end compiler 620 processes the intermediate representation 615, and generates therefrom machine code 625. *Schmidt,* para. [0040], lines 5–8. "Profile data is then gathered from executing the machine code from a set of sample inputs." *Schmidt,* para. [0039], lines 8–10.

B. The combination of Goebel and Schmidt does not render independent claims 1, 41, and 50 and dependent claims 3-10, 16, 17, 43-45, and 53 obvious

Claims 1, 3-10, 16, 17, 41, 43-45, 50, and 53 stand rejected under 35 USC § 103(a) as being obvious over the combination of Goebel and Schmidt.

When combining prior art elements to establish a prima facie case of obviousness, the MPEP requires a factual finding "*...that the prior art include* ***each*** *element claimed....*" M.P.E.P. § 2143 (A)(1). "All words in a claim must be considered in judging the patentability of that claim against the prior art." M.P.E.P. § 2143.03.

Independent claim 1 recites, in pertinent part,

> **generating data that indicates performance of the intermediate code when the intermediate code is executed** with the external execution input;

Appellants respectfully submit that the combination of Goebel and Schmidt fails to teach or suggest generating data that indicates performance of intermediate code when the intermediate code is executed.

The Office Action cites col. 4, line 66 to col 5, line 6; col. 6, lines 61-66; and col. 7, lines 62-65 of Goebel as teaching the above feature of claim 1. *Office Action mailed 06/08/09,* page 3. However, these citations suggest that **Goebel does not teach**

generating data that indicates performance of the intermediate code when the intermediate code is executed.

As cited by the Office Action, Goebel states, "An execution profile is a collection of data, gathered while a program executes ... [and] used to analyze the program's performance." *Goebel*, col. 4, line 66 to col. 5, line 6. However, Goebel fails to teach or suggest, an execution profile or any other data generated *indicating performance* ***of the intermediate code*** *when the intermediate code is executed.* The Office Action cites,

> Next, an 'adjustment instrumentation' procedure 505 conditionally removes, adds, or ignores profiling instructions in the **intermediate representation**. These profiling instructions are used to gather and save profile data relating to the execution history and/or the memory cache performance of the **executing program**. *Goebel*, col. 7, lines 60-65.

However, the next sentence of the cited paragraph discloses,

> That is, when the **binary executable is executed** by the computer, the profiling procedures measure relevant characteristics of the **executing program** and store this information as profile data. *Goebel*, col. 7, line 65 to col. 8, line 2.

Therefore, the cited "executing program" in col. 7, line 65 that generates the "profile data" is a binary executable–not an intermediate representation. Consequently, Goebel discloses generating profile data **of the binary executable** when the binary executable is executed, but Goebel fails to teach or suggest generating data that indicates performance **of the intermediate code** when the intermediate code is executed.

The Office Action further cites,

> The intermediate representation optimizer segment 305 of the re-optimzing compiler 300 can also process profile information generated during **execution of an instrumented binary executable** to determine which portions of the binary executable most need to be optimized. *Goebel*, col. 6, lines 61–66.

Here again, Goebel only discloses generating data during **execution of a binary executable**.

Lastly, the Office Action cites,

> The intermediate representation optimizer segment 305 and the code generator segment 307 also propogate portions of their internally collected **symbol and alias information** as annotation to the resulting binary module.... *Goebel*, col. 6, lines 35–39.

Even if intermediate representation optimizer segment 305 could be interpreted as "intermediate code" and "propagate" could be interpreted as "generating," (Appellants do not concede this interpretation) the "symbol and alias information" is data structure information not "data that indicates performance of the intermediate code." Therefore, this portion of Goebel fails to teach or suggest **generating data indicating performance**, and certainly does not teach or suggest generating data that indicates performance of the intermediate code when the intermediate code is executed.

In short, Goebel only discloses **a binary executable** generating data during execution. Thus, Goebel fails to teach or suggest generating data that indicates performance of the intermediate code when the intermediate code is executed.

Similarly, Schmidt also fails to teach or suggest generating data that indicates performance of the intermediate code when the intermediate code is executed. In fact, Schmidt recites,

> Back-end compiler 120 also includes a profiler 124 that is used to obtain profile data 126 **when the machine code 125 is run with a set of sample inputs**. As used herein, the term **"sample inputs" means inputs that simulate real-world execution of the machine code** in its intended environment. (*Schmidt*, para. [0030], emphasis added)
>
> **Profile data is then gathered from executing the machine code** from a set of sample inputs. (*Schmidt*, para. [0039], emphasis added)
>
> Once the profiler 624 has **generated profile data 626 for the machine code** that is output by the machine code emitter 622, the profile data 626 may be examined to determine how often each target method that corresponds to a virtual method call was actually invoked **during the sample execution of the machine code**. (*Schmidt*, para. [0042], emphasis added)
>
> Next, a user runs the instrumented program (i.e., **machine code** generated on the first pass of the back-end compiler) on sample inputs to gather profile data (step 730). (*Schmidt*, para. [0044], emphasis added)

Accordingly, these portions of Schmidt disclose that only **machine code** 125 or 625 is executed. Referring to FIG. 6, Schmidt differentiates between source code 105, intermediate representation 615, and machine code 625; however, Schmidt only discloses

that machine code 125 (FIG. 1) or machine code 625 (FIG. 6) is executed to generate profile data 126 or 626. As such, Schmidt also fails to teach or suggest generating data that indicates performance of the intermediate representation 115 or 615 when the **intermediate code is executed**.

Consequently, the combination of Goebel and Schmidt fails to teach or suggest all elements of claim 1, as required under M.P.E.P. § 2143.03. **Independent claims 41, and 50 include similar nonobvious elements as independent claim 1.** Accordingly, Appellants request that the instant §103(a) rejections of claims 1, 41, and 50 be withdrawn.

As to dependent claims 3-10, 16, 17, 43-45, and 53, if an independent claim is allowable, then any claim depending therefrom is also allowable. *See, e.g.,* MPEP § 2143.03; *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988). As discussed above, independent claims 1, 41, and 50 are in condition for allowance. Appellants submit that claims 3-10, 16, 17, 43-45, and 53 are therefore allowable by virtue of their dependence on allowable independent claims, as well as by virtue of the features recited in the claims. Appellants respectfully request withdrawal of the rejections and allowance of these claims.

C. The combination of Goebel and Schmidt does not render independent claim 18 and dependent claims 19 and 20 obvious

Claims 18-20 stands rejected under 35 USC § 103(a) as being obvious over the combination of Goebel and Schmidt.

When combining prior art elements to establish a prima facie case of obviousness, the MPEP requires a factual finding "*...that the prior art include* ***each*** *element claimed....*" M.P.E.P. § 2143 (A)(1). "All words in a claim must be considered in judging the patentability of that claim against the prior art." M.P.E.P. § 2143.03.

Independent claim 18 recites, in pertinent part,

> providing the intermediate code to a profiler that executes the intermediate code based on external execution input and **generates annotated intermediate code based on the performance of the executed intermediate code when the intermediate code is executed** with the external execution input;

Appellants respectfully submit that the combination of Goebel and Schmidt fails to teach

or suggest generating annotated intermediate code based on the performance of intermediate code **when the intermediate code is executed**.

The Office Action cites col. 6, lines 45–67; col. 7, lines 3–52; and element 407 of Goebel as teaching the above feature of claim 18. *Office Action mailed 06/08/09*, page 9. The Office Action references the phrases "process profile information" and "generated during execution" from col. 6, lines 63–64. *Office Action* mailed 06/08/09, page 9. However, the full sentence reads,

> The intermediate representation optimizer segment 305 of the re-optimzing compiler 300 can also *process profile information generated during* **execution of an instrumented binary executable** to determine which portions of the binary executable most need to be optimized. *Goebel*, col. 6, lines 61–66.

Therefore, Goebel only discloses processing profile information generated during execution of a binary executable, but Goebel fails to teach or suggest executing intermediate code. Since Goebel only discloses **executing a binary executable**, Goebel does not teach or suggest providing the intermediate code to a profiler that **executes the intermediate code** and generates annotated intermediate code based on the performance of the **executed intermediate code**.

The Office Action also references the phrase "processes…annotation information…was generated during compilation" found in Goebel col. 7, lines 11–13. *Office Action mailed 06/08/09*, page 9. The entire passage reads,

> The dissassembler procedure 403 inputs and converts the binary executable into a disassembled representation – an intermediate representation. The dissassembler procedure 403 also *processes* any supplied *annotation information* to approximate the compiler's state that was *generated during the compilation* **of the source**. *Goebel*, col. 7, lines 8–13.

First, one skilled in the art would recognize that **compiling code** is different from **executing code**. The drafter of the document repeatedly used the word "execute" to invoke the conventional definition of the word "execute" distinct from the word "compiling." Accordingly, it is unreasonable to interpret "compilation of the source" as being equivalent to "execution of the source."

Second, the alleged "code" being compiled in the passage is not the intermediate representation. To be sure, the passage states that the product or result of step 403 is the

intermediate representation. *Goebel*, col. 7, lines 10–13. Since the sentence in *Goebel*, col. 7, lines 8–10, is still talking about the process of step 403, and because step 403 has not been completed, the intermediate representation has not yet been created. Necessarily, the intermediate representation cannot be compiled or executed by step 403.

Third, "the compilation of the source" is almost certainly referring to "source code"—not an intermediate representation. In fact, page 11 of the Office Action mailed June 8, 2009, references *Goebel*, col. 7, lines 12-13 to assert that Goebel discloses the feature of claim 21 reciting, "producing further modified intermediate and machine code **based upon the source code**." The Office Action uses the same sentence to assert that an action is **based upon the execution of intermediate code for claim 18** (page 9 of the Office Action mailed June 8, 2009), but then uses the same sentence to assert an action is **based upon compilation of the source code** (page 11 of the Office Action mailed June 8, 2009). By doing this, the Office Action is contorting the reference to stitch together elements of the prior art in an unreasonable and inconsistent manner in an attempt to shoehorn the inconsistent interpretations into the Appellants' claims. When the Office Action gleans information exclusively from the Appellants' disclosure and uses the information as a roadmap for finding prior art references, the Office Action is engaging in Impermissible Hindsight. *See* M.P.E.P. 2145.X.A.

For the above reasons, Goebel only discloses **executing a binary executable** and **compiling source code**. Thus, Goebel fails to teach or suggest generating annotated intermediate code based on the performance of intermediate code **when the intermediate code is executed**.

Schmidt also fails to teach or suggest executing intermediate code for the same reasons outlined in Section VII.B above.

Consequently, the combination of Goebel and Schmidt fails to teach or suggest all elements of claim 18, as required under M.P.E.P. § 2143.03. Accordingly, Appellants request that the instant §103(a) rejections of claim 18 be withdrawn.

As to dependent claims 19 and 20, if an independent claim is allowable, then any claim depending therefrom is also allowable. *See, e.g.,* MPEP § 2143.03; *In re Fine*, 837 F.2d 1071 (Fed. Cir. 1988). As discussed above, independent claim 18 is in condition for allowance. Appellants submit that claims 19 and 20 are therefore allowable by virtue of

their dependence on allowable independent claims, as well as by virtue of the features recited in the claims. Appellants respectfully request withdrawal of the rejections and allowance of these claims.

VIII. CONCLUSION

Given the above arguments supporting patentability, Appellants believes all claims on appeal are in condition for allowance. If the undersigned attorney has overlooked a teaching in any of the cited references that is relevant to allowance of the claims, the Examiner is requested to specifically point out where such teaching may be found. Further, if there are any informalities or questions that can be addressed via telephone, the Examiner is encouraged to contact the undersigned attorney at (206) 292-8600.

Charge Deposit Account

Please charge our Deposit Account No. 02-2666 for any additional fee(s) that may be due in this matter, and please credit the same deposit account for any overpayment.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: Oct. 26, 2009

/Cory G. Claassen/
Cory G. Claassen
Attorney for Appellant(s)
Registration No. 50,296
Phone: (206) 292-8600

APPENDIX A — CLAIMS

1. (Previously Presented) A method comprising:

receiving source code;

transforming the source code to intermediate code;

executing the intermediate code based on external execution input;

generating data that indicates performance of the intermediate code when the intermediate code is executed with the external execution input; and

producing machine code based on the data and the intermediate code.

2. (Cancelled).

3. (Previously Presented) The method of claim 1, wherein executing the intermediate code comprises simulating execution of the intermediate code.

4. (Previously Presented) The method of claim 1, wherein generating the data regarding the performance of the executed intermediate code comprises generating a performance profile.

5. (Previously Presented) The method of claim 4, wherein generating the data regarding the performance of the executed intermediate code further comprises annotating the intermediate code based, at least in part, on performance profile data.

6. (Original) The method of claim 5, wherein annotating the intermediate code comprises concatenating data structures that include the performance profile data to intermediate code to embed the performance profile data into the intermediate code.

7. (Original) The method of claim 5, wherein annotating the intermediate code comprises:

generating a file that includes the performance profile data; and

mapping the performance profile data to corresponding portions of intermediate code.

8. (Previously Presented) The method of claim 5, wherein producing machine code based

on the data and intermediate code includes providing the annotated intermediate code to a compiler, wherein the compiler produces the machine code based on annotated intermediate code.

9. (Original) The method of claim 5, wherein the performance profile data comprises one or more of branch statistics, loop statistics and function invocation statistics.

10. (Original) The method of claim 8, wherein the machine code executes faster than the intermediate code.

11. – 15. (Cancelled).

16. (Previously Presented) The method of claim 1, further comprising:
receiving the external execution input; and
using the external execution input to execute the intermediate code.

17. (Original) The method of claim 1, wherein the data comprises one or more of plain-text format, binary representations, database maps, and character delimited proprietary format.

18. (Previously Presented) A method comprising:
transforming source code into intermediate code;
providing the intermediate code to a profiler that executes the intermediate code based on external execution input and generates annotated intermediate code based on the performance of the executed intermediate code when the intermediate code is executed with the external execution input;
receiving from the profiler the annotated intermediate code; and
transforming the annotated intermediate code into machine code.

19. (Original) The method of claim 18, wherein the annotated intermediate code is annotated to include one or more of branch statistics, loop statistics and function invocation statistics.

20. (Original) The method of claim 18, wherein providing the intermediate code to a profiler comprises providing the intermediate code to a virtual machine.

21. – 40. (Cancelled).

41. (Previously Presented) An article of manufacture comprising:

a computer readable storage medium including thereon sequences of instructions that, when executed, cause an electronic system to:

receive source code;

produce intermediate code based on the source code;

execute the intermediate code based on external execution input;

generate performance data that indicates performance of the intermediate code when the intermediate code is executed with the external execution input; and

produce machine code based on the intermediate code and the performance data.

42. (Cancelled).

43. (Previously Presented) The article of manufacture of claim 41, wherein the sequences of instructions that, when executed, cause the electronic system to generate the data regarding the performance of the executed code comprise sequences of instructions that, when executed, cause the electronic system to generate a performance profile.

44. (Original) The article of manufacture of claim 43, wherein the sequences of instructions that, when executed, cause the electronic system to cause the executed code to be modified based, at least in part, on the data comprise sequences of instructions that, when executed, cause the electronic system to annotate the intermediate code based, at least in part, on performance profile data.

45. (Previously Presented) The article of manufacture of claim 44, wherein the computer readable storage medium further comprises sequences of instructions that, when executed, cause the electronic system to provide the annotated intermediate code to a compiler, wherein the

compiler transforms the annotated intermediate code into machine code.

46. – 49. (Cancelled).

50. (Previously Presented) A system comprising:

a processor;

a dynamic random access memory coupled with the processor; and

an article of manufacture comprising a computer readable storage medium including thereon sequences of instructions that, when executed, cause an electronic system to:

receive source code;

produce intermediate code based on the source code;

execute the intermediate code based on external execution input;

generate data that indicates performance of the intermediate code when the intermediate code is executed with the external execution input; and

produce machine code based on the data and the intermediate code.

51. – 52. (Cancelled).

53. (Previously Presented) The system of claim 50, wherein the computer readable storage medium further comprises sequences of instructions that, when executed, cause the electronic system to:

receive external execution input; and

use the external execution input to execute the intermediate code.

APPENDIX B— EVIDENCE

[No Evidence is Entered]

APPENDIX C — RELATED PROCEEDINGS

[No Related Proceedings]